УДК 004.021 + 004.623 + 004.421

РАЗРАБОТКА ПРОГРАММНОГО КОМПЛЕКСА ГЕНЕРАЦИИ ВОПРОСОВ ПО ЗАДАННЫМ СУБЪЕКТАМ ПРИ ПОМОЩИ СЕМАНТИЧЕСКОЙ СЕТИ

М. Д. Андреичев[1], А. А. Ференец[2]

[1-2] Высшая школа информационных технологий и интеллектуальных систем Казанского (Приволжского) федерального университета

[1] andreichev.m@mail.ru, [2]ist.kazan@gmail.com

Аннотация

Представлен подход к автоматическому построению вопросов для тестов или викторин при помощи графа знаний DBPedia. Выбранный граф знаний имеет около 5 млн. сущностей и дает возможность делать запросы к семантической сети при помощи языка SPARQL. В статье представлены алгоритм, основные запросы к графу знаний для построения вопросов и нестандартный подход к поиску сущностей.

Ключевые слова*: семантическая сеть, генерация вопросов, связанные данные, онтология, граф знаний, RDF, SPARQL, DBPedia*

Введение

В сфере образования и обучения большую роль играют тесты. Например, в 2019 году тест ЕГЭ по истории состоит из двух частей, первая из которых включает в себя 19 вопросов с вариантами ответов. Также существуют специальные викторины для облегчения запоминания материала. При этом для 2019 года утверждено 15 общеобразовательных предметов, по которым проводится ЕГЭ. Для каждого предмета создаются десятки вариантов, что соответствует сотням вопросов для каждого предмета. Как правило, каждый вопрос в тесте или викторине строится вокруг какого-то одного понятия и предполагает простую структуру, что может означать, что вопросы можно генерировать автоматически, если иметь базу данных понятий и их соотношений. Было выдвинуто предположение, что возможно строить вопросы с вариантами ответа при помощи семантических сетей. В то же время важно не

просто создавать вопросы, но и иметь возможность выбирать тематику этих вопросов, то есть необходимо иметь возможность выбирать субъекты, по которым будут предложены вопросы.

Таким образом, была поставлена цель – разработать программный комплекс для построения вопросов по субъекту, найденному или определенному при помощи семантической сети, с сопутствующим функционалом.

1. Графы знаний

Для решения текущей задачи (создания вопросов компьютером) требуется компьютерное представление информации (приведение к определенной структуре) в виде семантической сети (рис. 1). Одно из таких решений разработано и утверждено Консорциумом Всемирной паутины: это структура (модель представления данных) — Resource Description Framework (RDF) [1].

Основная концепция RDF — триплет. Это набор «субъект, предикат,

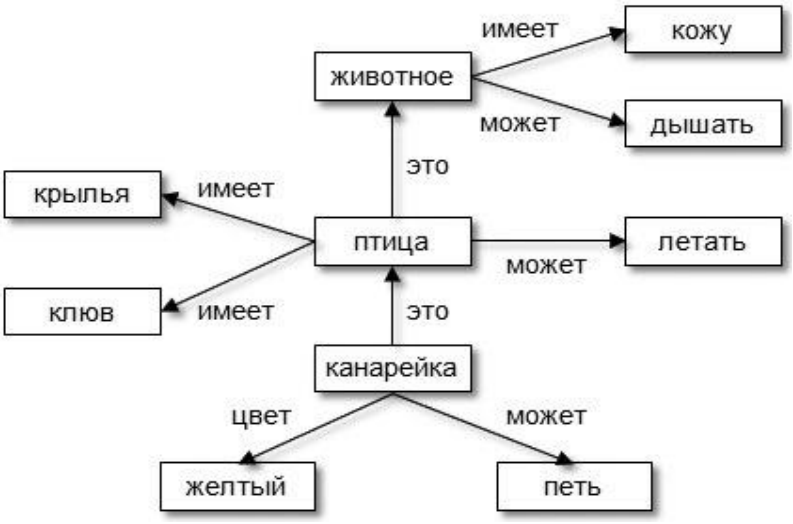


Рис. 1 — Семантическая сеть

объект» (например: дельфин – живет в – вода). В структуре RDF-ресурсом могут быть представлены любой объект реального мира или абстрактное понятие. Информацию в RDF можно просматривать в виде графа или утверждений и хранить в различных форматах: JSON, XML, N-Triples, Turtle, N3. Триплеты из RDF-документа объединяются в RDF-граф.

Связанные данные (linked data) играют все более важную роль в расширении интеллектуальных возможностей поиска в интернете. Связанные данные — это семантическая сеть, удовлетворяющая принципам, утвержденным Тимом Бёрнс-Ли [2]:

1) Использование URI в качестве идентификатора;
2) Предоставление URI, доступного по HTTP;
3) Предоставление доступа по стандартам SPARQL и RDF;
4) Предоставление по URI ссылки на другие URI.

В интернете имеются ресурсы связанных данных, в которых информация представлена в виде RDF. Одними из крупных ресурсов, в которых данные открыты в публичном доступе (linked open data или knowledge graph — граф знаний), являются Wikidata, DBPedia [3]. Сравнение Wikidata и DBPedia представлено в таблице 1.

DBpedia содержит множество ссылок на другие наборы данных в облаке LOD, таких, как Freebase, OpenCyc, GeoNames и другие. DBpedia широко использовалась в исследовательском сообществе Semantic Web, но также стала актуальной в коммерческих установках: например, такие компании, как BBC и New York Times, используют DBpedia для организации своего контента.

DBpedia берет свое начало в академических исследованиях, но также постоянно развивается благодаря сотрудничеству академических исследований и промышленности. Английская версия базы знаний DBpedia описывает 4,58 миллиона сущностей, из которых 4,22 миллиона классифицированы в последовательной онтологии, включая 1 445 000 человек, 735 000 мест (в том числе 478 000 населенных мест), 411 000 творческих работ (в том числе 123 000 музыкальных альбомов, 87 000 фильмов и 19 000 видеоигр), 241 000 организаций (включая 58 000 компаний и 49 000 учебных заведений), 251 000 видов и 6000 болезней. Это те данные, вокруг которых планируется строить запросы [4].

Таблица 1 — DBPedia и Wikidata

	Wikipedia	DBpedia	Wikidata
Сайт	wikipedia.org	dbpedia.org	www.wikidata.org
Метод генерации	Вручную/ заполнение сообществом	Автоматически или полу-автоматически	Полу-автоматически и заполнение сообществом (около 20000 редакторов)
Экземпляры, которые представляют объекты реального мира	——	4 298 433	18 697 897
Покрытие классов (сущностей >1)	——	88 %	41 %
Плюсы	Свободный текст /Легкость доступа и вклада	Большое количество интеграций в облаке LOD.	Качество (точность), редактирование сообществом, рост
Поддержка языков	294 (май 2019)	>125 (март 2019)	>358 (март 2019)

Предприятия, такие, как Apple (через Siri), Google (через Freebase и Google Knowledge Graph) и IBM (через Watson), и особенно соответствующие их проекты с высокой степенью видимости, связанные с искусственным интеллектом, получили огромную пользу от вклада DBpedia.

DBpedia также остается основой для академических занятий в областях проектирования онтологий, искусственного интеллекта, машинного обучения, обработки естественного языка и многого другого. Проект DBPedia оказался полезным для экспертов, ведущих проекты в Google, Microsoft, Facebook, Oracle, IBM, Apple и многих других.

2. Существующие решения

В рамках решения поставленной задачи был произведен обзор существующих решений, затрагивающих тематику подготовки тестов и викторин. Были найдены сервисы, которые способны создавать базу вопросов.

Многие из сервисов дают возможность проводить викторину онлайн, но вопросы в них не генерируются при помощи семантических сетей автоматически.

Также найдено приложение для мобильных устройств – программа-викторина clover quiz [5]. Она использует базу вопросов, сгенерированную с использованием семантических сетей. К сожалению, база вопросов фиксирована и не позволяет делать выгрузку. Создатель программы ссылается на программу linkeddata trivia [6]. Эта система способна сгенерировать вопрос. Но на это у приложения может уходить больше минуты. Тема вопроса не может быть задана, и вопрос строится вокруг неизвестного понятия, что делает использование этого инструмента крайне затруднительным для создания тестов и викторин на определённые темы. Для запуска этого приложения требуется запуск двух микросервисов, один на java, другой на javascript. Код этого приложения находится в открытом доступе.

При генерации вопроса в приложении linkeddata trivia к графу знаний выполняется большое количество SPARQL-запросов (около 30). Вопрос строится вокруг случайно взятой сущности. В базе более 4 миллионов реально существующих объектов, и получая случайную сущность, маловероятно сгенерировать вопрос, на который пользователь сможет ответить. Получившиеся вопросы представлены в таблице 2.

Таблица 2 – Сгенерированные вопросы

Вопрос	What is the location of 63 Building?			
Варианты ответов	Yeouido	Las Vegas	Nkawkaw	Eli, Mateh Binyamin
Время	20101,9 мс			
Вопрос	What is the location of San Bernardino Pass?			
Варианты ответов	Switzerland	Miholjsko	Svilići	Leira, Ørsta
Время	12273,5 мс			
Вопрос	What is the vein of Heart?			
Варианты ответов	Pulmonary vein	Angular vein	Deep dorsal vein of clitoris	Median antebrachial
Время	2334,2 мс			
Вопрос	What is the death place of William Warfield?			
Варианты ответов	Alumim	Chicago	Les Herbiers	Takoundou
Время	23447,1 мс			
Вопрос	What is the family of Pleuronodoceratidae?			
Варианты ответов	Xenodiscaceae	Cystoseira foeniculacea	Angaria rugosa	Scopula umbelaria
Время	23447,1 мс			

Приложение linkeddata trivia состоит из двух микросервисов. Один называется backend-core [7]. Он написан на java и запускается при помощи фреймворка Grizzly. Его роль состоит в том, чтобы получить наиболее часто встречающиеся предикаты объекта.

Другой микросервис запускается при помощи фреймворка Node js. Основной код построения вопроса находится на нем. Построение вопроса

приложение выполняет по несколько попыток (в среднем их требуется около 10). На каждую попытку приходится около 10 SPARQL-запросов. Попытка срывается в результате неподходящего ответа от DBPedia, и алгоритм запускается заново (вызывается catch метод в последовательности Promise javascript). Далее будет рассмотрена следующая последовательность построения вопроса микросервисом.

1. Подобрать любую случайную сущность из DBPedia. Для подбора случайной сущности требуется в запросе SELECT указать случайное значение offset. В программе имеется файл classes_sorted.json, в котором хранится количество сущностей на граф знаний для каждого класса. Из этого файла берутся значения, в каких пределах должно быть загадано случайное число для генерируемого объекта.

2. Сделать запрос и получить rdfs:label объекта. В нем хранится литерал с подписью объекта — текст на каком-то языке. Литерал состоит из текста литерала в кодировке Unicode и метки языка в формате RFC 3066.

3. Получить предикаты объекта (те, которые имеют литерал).

4. Сделать запрос на микросервис backend-core, получить наиболее часто встречающиеся предикаты объекта. Они хранятся в json-файле на сервере. Backend-core делает SPARQL-запрос для определения типа запрашиваемой сущности. В зависимости от типа из собственной базы возвращает значение.

5. Сделать запрос об информации о выбранном предикате, по которому будет строиться запрос. Выясняется литерал и rdfs:range (тип) объекта. На этом шаге часто прерывается алгоритм и запускается заново из-за того, что отсутствуют нужные предикаты у объекта.

6. Получить альтернативные варианты ответов. Если варианты числовые — запросов не делать и альтернативные варианты подобрать программно.

При необходимости создания вопроса на определённую тематику пользователю приходится несколько раз запускать генератор. Это происходит из-за случайного подбора предикатов, на которых требуется построить вопрос. Запросы можно усовершенствовать так, чтобы на этапе 3 заранее предусматривалось то, что будет нужно делать с субъектом на следующих этапах.

Были построены вопросы по классу dbo:Event (события) при помощи части кода системы linkeddata trivia. Вопросы получились по различным событиям: военные конфликты, спортивные события, парады и другие. Вопросы, построенные системой linkeddata trivia, представлены в таблице 3.

Таблица 3 — Вопросы, построенные linkeddata trivia

Вопрос	Ответ
Кто является командиром блокады Кисо-Фукусима?	Такэда Сингэн
Что является частью военного конфликта операции «Радуга»?	Второй Интифады
Что является частью военного конфликта битве минимойке?	Американская Революционная Война
Предыдущее событие Кубка 2013 финал Дель Рей?	2012 Копа Дель Рей финал
Каково место военного конфликта Битва Salsu?	Река Chongchon

3. Требования к программному комплексу

Получившаяся система должна удовлетворять следующим требованиям:

1) Возможность поиска сущности по онтологии (не более 10 секунд на поиск по онтологии);

2) Возможность извлечения случайной сущности, подходящей по заданному учебному предмету. Предмет определяет классы, по которым извлекать случайные сущности;

3) Возможность извлечений сущности для определенной зоны, отмеченной на карте (для поиска сущности по карте);

4) Построение вопросов по сущности;

5) Построение альтернативных (неверных) вариантов ответа.

4. Архитектура и API системы

Система реализована на языке Java с применением фреймворка Spring Web MVC, который обеспечивает архитектуру паттерна Model View Controller (MVC). Для создания http-запросов используется библиотека Apache Jena. Для генерации вопросов частично используется модифицированный код linkeddata trivia.

Большая часть системы весьма тривиальна в реализации (модель с аннотациями Spring Data Jpa, контроллеры), поэтому рассматриваться в данной работе не будет. Исходный код приложения выложен в открытый доступ: https://github.com/NGdev1/QuizEngine. Архитектура представлена на рисунке 2.

PredicatesRequestsService отвечает за запросы извлечения подходящих триплетов для построения вопроса и за построение альтернативных вариантов ответа.

ClassesRequestsService отвечает за извлечение случайных сущностей или поиск сущностей по заданным критериям, например, за поиск мест в базе, входящих в зону поиска для создания вопросов из области географии.

SparqlService объединяет эти сервисы и выполняет не только запросы, но и имеет логику их обработки. При построении альтернативных вариантов ответа SparqlService запрашивает у AlternativeAnswersHandler, требуется ли делать запрос к базе знаний для получения альтернативных (неверных) вариантов ответа. Если нет — сразу получает их.

PredicatesRequestsService и ClassesRequestsService, используя библиотеку Apache Jena, которая в свою очередь использует HttpClient, в контексте программы названный SparqlHttpClient.Jena, делает http запрос на DBPedia SPARQL точку доступа [8] или любую другую, указанную в конфигурации системы.

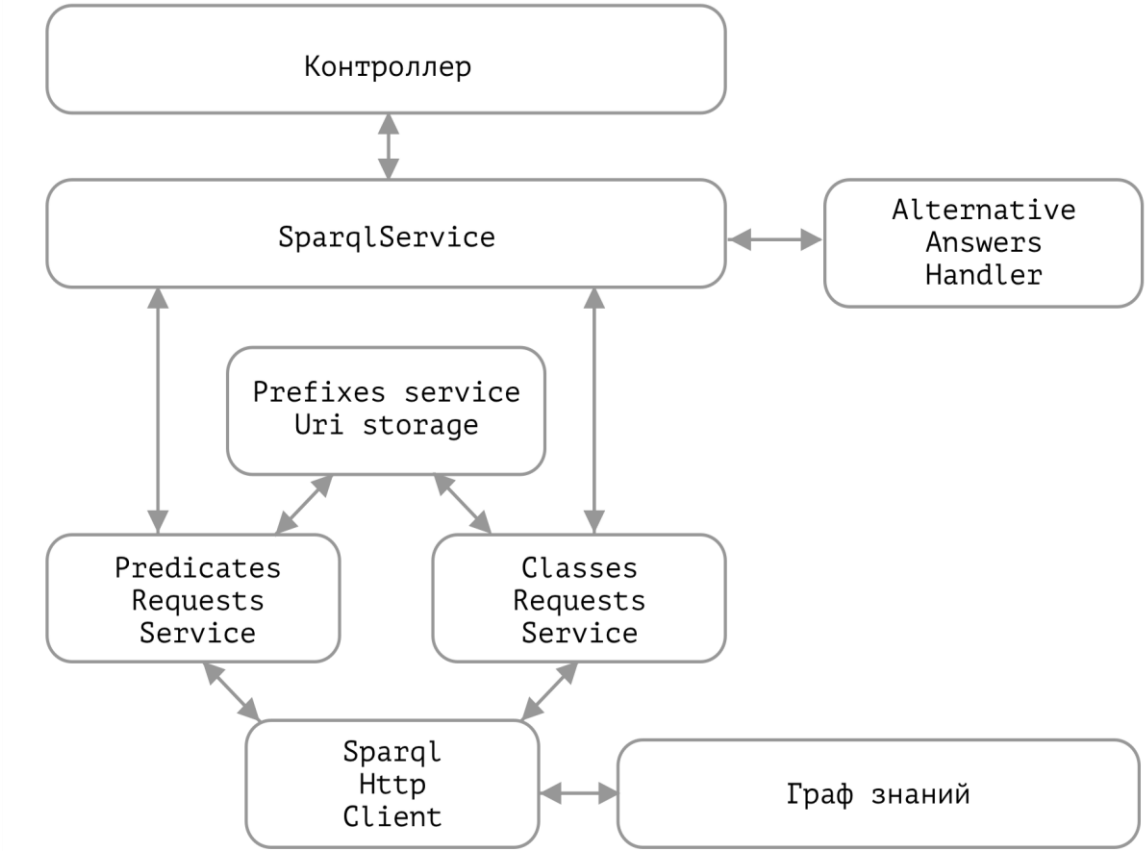


Рис. 2. — Архитектура

PrefixesService заменяет uri на сокращение или, наоборот, возвращает список префиксов с нужном формате для построения SPARQL-запроса.

Ниже приведены основные url для api системы, которые касаются исключительно генерации вопросов по заданным субъектам при помощи семантической сети.

1) Извлечение случайной сущности: /api/random-entity. В качестве параметра принимает класс сущности, по которой необходимо извлекать сущности. Для DBPedia список классов перечислен по ссылке DBPedia Ontology [9]. Если класс — dbo:Place, то необходимо извлечь из онтологии место. В таком случае рассматривается параметр запроса region. В нем передается список координат той области, внутри которой производится поиск мест по онтологии. В region передаются 4 значения координат, разделенных запятыми.

2) Запрос /api/find-entity возвращает то же, но в результате несколько вариантов, в предыдущем запросе возвращается только один. Запрос так же

принимает параметры класс, регион, но еще и строку, по которой происходит поиск.

3) Запрос /quiz/{quizId}/add-question-with-entity возвращает список доступных вопросов, которые могут быть добавлены в тест. В качестве параметра необходимо передать URI сущности, по которой будут сгенерированы вопросы.

4) Запрос /api/alternative-answers возвращает альтернативные варианты ответов для вопроса. Принимаемые параметры — класс предиката корректного варианта ответа. Например, если в ответе фигурирует страна, альтернативными вариантами должны быть другие страны. Еще запрос принимает параметр — корректный вариант ответа. Если корректный вариант — число или дата, никаких запросов к DBPedia не будет, и альтернативные варианты ответа (неверные) будут построены программно.

5) Редактирование вопроса, POST запрос: /quiz/{quizId}/question/{questionId}.

6) Удаление вопроса, DELETE запрос: /quiz/{quizId}/question/{questionId}.

7) Добавление вопроса в систему (финальный этап), POST запрос: /quiz/{quizId}/add-question/.

Для запросов к DBPedia используется библиотека Apache Jena, и приложение написано полностью на Java, опираясь на алгоритм linkeddata trivia.

Для извлечения литерала сущности (подписи на определенном языке) требуется задать язык в фильтре запроса. Многие ресурсы онтологии DBPedia, даже из тех, которые ссылаются на объекты реального мира и находятся в России, не имеют подписи на русском языке. Было решено строить каждый запрос с приоритетом языка. Если имеется подпись на русском языке — выводить подпись на русском, если нет — на английском. Это достигнуто тем, что фильтр подписи на русском ставится как OPTIONAL (не обязательный). Подпись на английском является обязательной, так как она имеется у каждого объекта. Далее переменные языков объединяются в одну методом COALESCE. Форма функции COALESCE возвращает значение RDF-члена первого выражения, которое оценивается без ошибок. В системе должны быть указаны два языка: язык с приоритетом 1 и язык с приоритетом 2. Язык с приоритетом 1 по умолчанию задан как русский (ru), а с приоритетом 2 — английский (en).

Пример использования функции COALESCE для извлечения литерала с приоритетом двух языков:

```
select ?entity, coalesce(?labelLang1, ?labelLang2) as ?label where {
 OPTIONAL {
  ?entity rdfs:label ?labelLang1 .
  FILTER(langMatches(lang(?labelLang1), "ru")) .
 }
 ?entity rdfs:label ?labelLang2 .
  FILTER(langMatches(lang(?labelLang2), "en")) .
}
```

5. Алгоритм

Чтобы пользователь имел возможность строить вопросы и по заданному субъекту, и по случайному субъекту, алгоритм должен предусматривать различные способы генерации. Основные способы построения вопросов: по поиску субъекта; случайному субъекту, определенному по классу; случайному субъекту, определенному по тематике вопросов. Алгоритм построения вопросов имеет три ответвления, основанные на способах генерации.

В ходе работы был создан и полностью реализовал алгоритм, способный быстрее делать то, что делает библиотека linkeddata trivia с более гибким применением (добавлена возможность поиска ключевой сущности для построения вопроса). Соответствующий алгоритм представлен на рисунке 3. Опорными пунктами являются подбор субъекта и построение вопросов по подходящим триплетам субъекта. При подборе субъекта по поиску в семантической сети DBPedia было решено использовать систему Google Custom Search для ускорения поиска.

6. Извлечение сущности

Извлечение сущности, по которой будет построен вопрос, — первичная задача системы при построении вопроса. Ниже приведен SPARQL-запрос, по которому извлекается сущность при поиске в базе DBPedia по запросу «Пушкин».

```
select distinct ?entity ?label where {
 ?entity a dbo:Person .
```

```
  ?entity rdfs:label ?label .
  FILTER contains(?label, "Пушкин") .
  FILTER(langMatches(lang(?label), "ru") || langMatches(lang(?label), "en"))
} limit 100
```

Префиксы — сокращения uri. В данном запросе использовались распространенные сокращения dbo, rdfs. На поиск ушло 54 секунды. Это очень долго, и такой запрос представляет нагрузку для системы DBPedia.

Альтернативный способ поиска по DBPedia. Ресурсы в DBPedia доступны как веб страницы по ссылке http://dbpedia.org/page/name, где name — имя ресурса. А URI ресурсов представлен так: http://dbpedia.org/resource/name. Поиск по веб-страницам любого сайта доступен при помощи поисковых систем. Было решено использовать поисковую систему Google для поиска необходимой страницы, после чего обращаться к необходимому ресурсу в базе знаний.

Система Custom Search JSON API позволяет настраивать программный поиск и отображения результатов из поиска Google. С помощью этого API была решена задача поиска по онтологии в виде поиска веб страниц отображаемых ресурсов онтологии. Результаты поиска обрабатывались в формате JSON.

Custom search engine ID — используется, чтобы указать пользовательскую поисковую систему, которая необходима для выполнения поиска. Пользовательская поисковая система (Custom Search Engine) должна быть создана с помощью панели управления (https://cse.google.com). В этой панели управления доступны настройки сайтов, на которых будет выполняться поиск, и сайты, исключенные из поиска. Пришлось исключить из поиска сайты http://eo.dbpedia.org, http://cs.dbpedia.org и другие для того, чтобы исключить из поиска некоторые нерабочие ссылки для системы генерации вопросов (те, которые не ссылаются на ресурсы онтологии DBPedia).

Ограничение использования бесплатной версии системы Google CSE -— 10000 запросов в день.

Пользовательская поисковая система реализована на стороне клиента. По запросу за 0.34 секунды были найдены подходящие результаты. Первый результат по запросу «Пушкин»: http://dbpedia.org/page/Alexander_Pushkin. Полученный url требуется преобразовать, заменив /page на /resource:

```
entity = entity.replace('/page', '/resource').
```

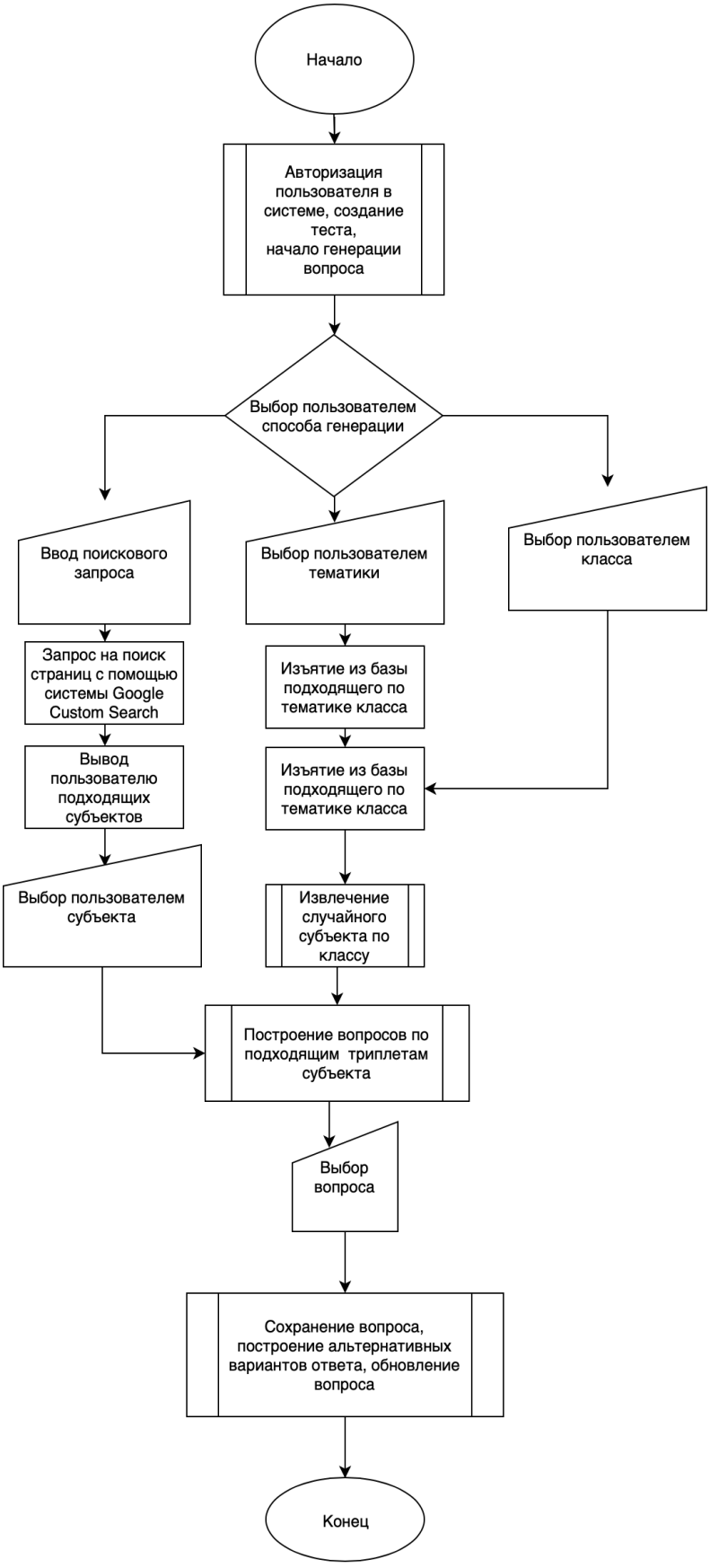


Рис. 3. — Алгоритм построения вопросов

Система получает Uri ресурсов семантической сети DBPedia за необходимое время по поисковому запросу.

Далее будет рассмотрена ситуация, в которой требуется извлечь сущность для региона, определенного по карте.

В первую очередь необходимо получить координаты, выбрав их на карте. В системе был выбран сервис Яндекс-карт. В параметрах Ajax-запроса используется следующий код: *myMap.getBounds().toString()*, где myMap — Яндекс-карта на веб-странице.

Если был выбран следующий регион: город Казань, SPARQL запрос, выполняемый на стороне сервиса, выглядит следующим образом:

```
PREFIX geopos: <http://www.w3.org/2003/01/geo/wgs84_pos#>
PREFIX georss: <http://www.georss.org/georss/>
select ?place coalesce(?labelLang1, ?labelLang2) as ?label where {
 ?place a dbo:Place .
 ?place geopos:lat ?lat .
 ?place geopos:long ?long .
 OPTIONAL {
  ?place rdfs:label ?labelLang1 .
  FILTER(langMatches(lang(?labelLang1), "ru")) .
 }
 ?place rdfs:label ?labelLang2 .
 FILTER (
  ?lat > 55.75866589150474 &&
  ?lat < 55.83602591889409 &&
  ?long > 48.98691801306621 &&
  ?long < 49.24441007849591 &&
  lang(?labelLang2) = "en"
 )
} LIMIT 1000
```

Результаты выполнения запроса следующие: Мост Миллениум (Казань), Казанский кремль, Кул-Шариф, Казань, Центральный (стадион, Казань), Баскет-холл (Казань), Горки (станция метро), Мечеть аль-Марджани, Храм Воздвижения Святого Креста (Казань), Суконная слобода (станция метро), Площадь Габдуллы

Тукая (станция метро) и другие. После того, как пользователь выберет подходящий субъект, программа построит по нему вопросы.

7. Построение вопросов

Имеется сущность, по которой системе нужно построить вопрос. Для построения вопроса требуется извлечь подходящие триплеты для этой сущности. Многие предикаты повторяются, и построение вопроса по этим предикатам не имеет смысла. Например, id ресурса онтологии на сайте Википедия. Чтобы исключить те предикаты, по которым не требуется строить вопрос, был реализован черный список предикатов, который хранится в PrefixesService.

Часть тех URI, которые вошли в черный список:

- "http://www.w3.org/1999/02/22-rdf-syntax-ns#type"
- "http://www.w3.org/2002/07/owl#sameAs"
- "http://www.w3.org/ns/prov#wasDerivedFrom"
- "http://dbpedia.org/ontology/wikiPageRevisionID"
- "http://dbpedia.org/ontology/wikiPageID"
- "http://dbpedia.org/ontology/wikiPageWikiLink"
- "http://dbpedia.org/ontology/wikiPageDisambiguates"
- "http://dbpedia.org/ontology/soundRecording"
- "http://www.w3.org/2000/01/rdf-schema#seeAlso"
- "http://www.w3.org/2000/01/rdf-schema#label" и другие.

Далее будет рассмотрены SPARQL-запросы, которые строятся сервисом PredicatesRequestsService. Первый запрос извлекает триплеты (субъект, предикат, объект), в которых субъектом является заданная сущность. Например, если вопросы строятся про Репина Илью Ефимовича, то в первом случае один из вопросов будет выглядеть так: Кто был под влиянием Репина Ильи Ефимовича? Ответ: Суриков, Василий Иванович. То есть основой вопроса в этом случае является заданная сущность. Из графа знаний извлекаются только те узлы, которые направлены от заданного узла (который ссылается на заданную сущность).

Ниже приведен сам первый SPARQL-запрос для извлечения триплетов для построения вопросов по заданному субъекту:

```
select DISTINCT coalesce(?subjectLabelLang1, ?subjectLabelLang2) as ?subjectLabel, ?predicate, coalesce(?predicateLabelLang1, ?predicateLabelLang2) as ?predicateLabel, ?object, coalesce(coalesce(?objectLabelLang1, ?objectLabelLang2), ?object) as ?objectLabel where {
  <http://dbpedia.org/resource/Ilya_Repin> ?predicate ?object.
  OPTIONAL {
   <http://dbpedia.org/resource/Ilya_Repin> rdfs:label ?subjectLabelLang1.
   FILTER(langMatches(lang(?subjectLabelLang1), "ru")).
  }
  <http://dbpedia.org/resource/Ilya_Repin> rdfs:label ?subjectLabelLang2.
  FILTER(langMatches(lang(?subjectLabelLang2), "en")).
  OPTIONAL {
   ?object rdfs:label ?objectLabelLang1.
   FILTER(langMatches(lang(?objectLabelLang1), "ru")).
  }
  OPTIONAL {
   ?object rdfs:label ?objectLabelLang2.
   FILTER(langMatches(lang(?objectLabelLang2), "en")).
  }
  OPTIONAL {
   ?predicate rdfs:label ?predicateLabelLang1.
   FILTER(langMatches(lang(?predicateLabelLang1), "ru")).
  }
  ?predicate rdfs:label ?predicateLabelLang2 .
  FILTER(langMatches(lang(?predicateLabelLang2), "en")).}
 limit 3000
```

В списке возвращаемых значений субъект (URI ресурса), его литерал, предикат (URI), его литерал, объект (URI), литерал объекта. Предикат нужен для построения в дальнейшем альтернативных вариантов ответа. Литерал объекта может отсутствовать, тогда URI объекта подставляется на место литерала. Это добавляет некоторые вопросы относительно заданного субъекта, когда

некоторые числовые значения для субъекта не имеют литерала ни на русском, ни на английском.

Второй запрос извлекает те триплеты, в которых объектом триплета является заданная сущность. В этом случает один из вопросов будет выглядеть так: Кто находился под влиянием Врубеля Михаила Александровича? Ответ: Репин, Илья Ефимович. Или такой вопрос: Кто является автором «Запорожцы» (картина)? Ответ: Репин, Илья Ефимович. То есть основой вопроса в этом случае являются те ресурсы, которые ссылаются на заданный ресурс. Из графа знаний извлекаются только те узлы, которые направлены к заданному узлу.

```
select DISTINCT ?subject, coalesce(?subjectLabelLang1, ?subjectLabelLang2) as ?subjectLabel, ?predicate, coalesce(?predicateLabelLang1, ?predicateLabelLang2) as ?predicateLabel, coalesce(?objectLabelLang1, ?objectLabelLang2) as ?objectLabel where {
  ?subject ?predicate <http://dbpedia.org/resource/Ilya_Repin> .
  OPTIONAL {
   <http://dbpedia.org/resource/Ilya_Repin> rdfs:label ?objectLabelLang1 .
   FILTER(langMatches(lang(?objectLabelLang1), "ru")) .
  }
  <http://dbpedia.org/resource/Ilya_Repin> rdfs:label ?objectLabelLang2 .
  FILTER(langMatches(lang(?objectLabelLang2), "en")) .
  OPTIONAL {
   ?subject rdfs:label ?subjectLabelLang1 .
   FILTER(langMatches(lang(?subjectLabelLang1), "ru")) .
  }
  ?subject rdfs:label ?subjectLabelLang2 .
  FILTER(langMatches(lang(?subjectLabelLang2), "en")) .
  OPTIONAL {
   ?predicate rdfs:label ?predicateLabelLang1 .
   FILTER(langMatches(lang(?predicateLabelLang1), "ru")) .
  }
  ?predicate rdfs:label ?predicateLabelLang2 .
  FILTER(langMatches(lang(?predicateLabelLang2), "en")) .}
 limit 3000
```

Distint означает, что возвращаемые после выполнения запроса значения не должны повторяться.

Резюмируя представленные запросы, выражаясь в терминологии триплетов, можно сказать, что первый запрос возвращает все предикаты и объекты по заданному субъекту, а второй запрос возвращает субъекты и предикаты по заданному объекту.

Результаты каждого запроса фильтруются через черный список предикатов, все результаты собирает сервис SparqlService и возвращает в виде массива TripleDto.

Построение шаблона для построения вопроса на английском достаточно просто. Из списка получившихся выше описанных триплетов построить список предлагаемых пользователю вопросов удается по шаблону:

What is the {triple.**predicate**Label} of {triple.**subject**Label}? ({triple.**object**Label}).

Было решено выбрать следующий шаблон для русского языка:

Кто, или что, или какой {triple.**predicate**Label} {triple.**subject**Label}? ({triple.**object**Label}).

Учитывая, что в базе более 400 млн триплетов, система способна сгенерировать разные вопросы, и около 300 млн из них могут не повторяться.

Получившийся список некоторых вопросов для заданного субъекта Казань (http://dbpedia.org/resource/Kazan) представлен в таблице 4. Всего для этого субъекта получилось 605 вопросов. Выбрав подходящий вопрос из списка, пользователь может добавить его в создаваемый тест.

8. Построение альтернативных вариантов ответа

Если необходимо предоставить альтернативный вариант ответа или если необходимо построить вопрос по случайному субъекту (как в linkeddata trivia), система выбирает случайную сущность по заданному классу. Для извлечения случайной сущности сперва необходимо получить случайное смещение (random offset). Смещение может быть указано от нуля до количества сущностей заданного класса. Чтобы извлечь количество сущностей для заданного класса, система выполняет следующий SPARQL-запрос:

Таблица 4 — Некоторые вопросы по ресурсу dbr:Kazan

Вопрос	Ответ
Кто, или что, или какой country Казань?	Россия
Кто, или что, или какой federal subject Казань?	Republic of Tatarstan
Кто, или что, или какой leader name Казань?	Метшин, Ильсур Раисович
Кто, или что, или какой May record low C Казань?	-6.5
Кто, или что, или какой May record high C Казань?	33.5
Кто, или что, или какой Jun record low C Казань?	-1.4
Кто, или что, или какой death place Кул Шариф (религиозный деятель)?	Казань
Кто, или что, или какой death place Девятаев, Михаил Петрович?	Казань
Кто, или что, или какой death place Вахитов, Мулланур Муллазянович?	Казань
Кто, или что, или какой birth place Никифоров, Николай Анатольевич?	Казань
Кто, или что, или какой birth place Шаляпин, Фёдор Иванович?	Казань
Кто, или что, или какой birth place Попов, Валерий Георгиевич?	Казань
Кто, или что, или какой stadium Чемпионат мира по футболу 2018?	Казань
Кто, или что, или какой city Чемпионат Европы по тяжёлой атлетике 2011?	Казань

```
select count(?obj) as ?count
where {
 ?obj a ?type .
 ?obj rdfs:label ?label
 FILTER(langMatches(lang(?label), LANGUAGE2)) .
}
```

Получившееся количество задается как верхняя граница случайного смещения. В следующем запросе извлекается случайная сущность:

```
select coalesce(?labelLang1, ?labelLang2) as ?label where {
 ?subject a ?type.
 OPTIONAL {
  ?subject rdfs:label ?labelLang1.
  FILTER(langMatches(lang(?labelLang1), LANGUAGE1)).
 }
 ?subject rdfs:label ?labelLang2.
 FILTER(langMatches(lang(?labelLang2), LANGUAGE2))
} OFFSET ?offset
LIMIT 1
```

При построении альтернативных вариантов ответа запрос к графу знаний не обязателен. Если значение верного ответа числовое или значение является датой или годом, альтернативные варианты ответа получаются программно.

Если необходимо извлечь альтернативный вариант ответа из графа знаний, система определяет rdfs:range для ресурса корректного варианта ответа. Rdfs:range объявляет класс или тип данных объекта. Выполняется следующий запрос:

```
select ?range where {
 < predicateUri> rdfs:range ?range.
}
```

После того, как система получает данные о классе сущности верного ответа, она получает случайную сущность по этому классу так, как это описано вышев данном разделе.

Система сохраняет класс верного ответа в базе данных вместе с составленным вопросом для того, чтобы при необходимости составитель теста

или викторины мог снова сгенерировать и поменять альтернативные варианты ответов.

9. Апробация. Генерация вопросов

Системой были построены вопросы для тестирования по теме «Живопись». Выбранные пользователем субъекты: Леонардо да Винчи, Репин, Илья Ефимович, Врубель, Михаил Александрович, Куинджи, Архип Иванович, Шишкин, Иван Иванович. Некоторые из построенных вопросов были выбраны для теста. Все вопросы были проверены вручную. На поиск субъектов уходило не более секунды. Компьютер, на котором происходило построение, имеет следующую конфигурацию: 4 гб озу, процессор 2,7 GHz Intel Core i5. Скорость соединения с интернетом: 30 мбит/с. На построение вопросов по заданному субъекту затрачено около пяти секунд. Построенные вопросы представлены в таблице 5.

Получившиеся вопросы переведены не полностью. Приоритет извлечения языковых литералов был установлен сначала на русский язык, и при его отсутствии извлекался литерал на английском языке. Те предикаты и субъекты семантической сети DBPedia, которые не имеют литерала на русском языке, остались на английском.

Таблица 5 — Построенные вопросы

Вопрос	Ответ
Кто, или что, или какой автор Мона Лиза?	Леонардо да Винчи
Кто, или что, или какой movement Репин, Илья Ефимович?	Реализм
Кто, или что, или какой movement Врубель, Михаил Александрович?	Символизм
Кто, или что, или какой training Врубель, Михаил Александрович?	Императорская Академия художеств
Кто, или что, или какой influenced by Куинджи, Архип Иванович?	Шишкин, Иван Иванович
Кто, или что, или какой автор Запорожцы (картина)?	Репин, Илья Ефимович
Кто, или что, или какой training Шишкин, Иван Иванович?	Московское училище живописи, ваяния и зодчества
Кто, или что, или какой death place Девятаев, Михаил Петрович?	Казань
Кто, или что, или какой death place Врубель, Михаил Александрович?	Санкт-Петербург
Кто, или что, или какой автор Демон сидящий?	Врубель, Михаил Александрович
Кто, или что, или какой works Шишкин, Иван Иванович?	Morning in a Pine Forest
Кто, или что, или какой автор Тайная вечеря?	Леонардо да Винчи
Кто, или что, или какой автор Портрет Джиневры де Бенчи?	Леонардо да Винчи

ЗАКЛЮЧЕНИЕ

Разработана система генерации вопросов для тестов и викторин по заданному субъекту, найденному в поиске или при помощи географической карты. С помощью разработанной системы были построены тесты, которые были предложены для прохождения десяткам студентов ВШ ИТИС КФУ. Ими были пройдены все тесты и отмечено, что все вопросы были понятны.

В созданной системе есть функционал, реализующий поиск субъектов для построения вопросов при помощи географической карты. Это дает возможность построения викторин и тестов по краеведению.

Клиентская часть системы может быть разработана для мобильных устройств так, что у пользователей будет возможность создавать обучающие тесты для себя и друг для друга.

Пользователи (школьники, студенты, преподаватели, другие), пользуясь функционалом системы, могут отправлять друг другу созданные тесты и таким образом повышать собственную эрудицию.

Созданную систему можно доработать, добавив в нее доступные графы знаний Wikidata. Это увеличит количество вопросов, которые способна предоставить система.

СПИСОК ЛИТЕРАТУРЫ

1. RDF Schema 1.1 // W3C. URL: https://www.w3.org/TR/rdf-schema/ (дата обращения: 25.03.2019).

2. *Tim Berners-Lee.* Linked Data. URL: https://www.w3.org/DesignIssues/LinkedData.html (дата обращения: 11.04.2019).

3. *Amrapali Zaveri*, University of Leipzig. Linked Data Quality of DBpedia, Freebase, OpenCyc, Wikidata, and YAGO. URL: http://www.aifb.kit.edu/images/c/ca/KG-Comparison-SWJ-Article.pdf (дата обращения: 05.04.2019).

4. Learn about DBPedia // About | DBPedia. URL: https://wiki.dbpedia.org/about (дата обращения: 10.04.2019).

5. *G. Vega-Gorgojo, Don Naipe*. Clover quiz: A mobile trivia game based on DBpedia data. URL: http://ceur-ws.org/Vol-1963/paper474.pdf (дата обращения: 11.04.2019).

6. Auto-generated trivia questions based on DBPedia data // n1try/ linkeddata-trivia. URL: https://github.com/n1try/linkeddata-trivia (дата обращения: 01.02.2019).

7. Linked Open Data Seminar 2016 – Knowledge Panel. // n1try/kit-lod16-knowledge-panel. URL: https://github.com/n1try/kit-lod16-knowledge-panel (дата обращения: 01.02.2019).

8. Virtuoso SPARQL Query Editor // DBpedia. URL: https://dbpedia. rg/ sparql (дата обращения: 20.03.2019).

9. Ontology Classes // DBPedia mappings. URL: http://mappings.dbpedia. org/server/ontology/classes/ (дата обращения: 27.03.2019).

DEVELOPMENT OF A SOFTWARE PACKAGE FOR GENERATING QUESTIONS FOR SPECIFIED SUBJECTS USING A SEMANTIC NETWORK

M. D. Andreichev [1], A. A. Ferenetz [2]

[1-2] Higher School for Information Technologies and Intelligent Systems of Kazan (Volga Region) Federal University

[1] andreichev.m@mail.ru, [2] ist.kazan@gmail.com

Abstract

An approach to automatically generating questions for tests or quizzes using the DBPedia knowledge graph is presented here. The selected knowledge graph has about 5 million entities. DBpedia SPARQL endpoint the ability to make queries to the semantic network using the SPARQL language. The algorithm, the basic queries to the knowledge graph for constructing questions, a non-standard approach to the search for entities are presented in this article.

Keywords*: semantic network, generation of questions, linked data, ontology, knowledge graph, RDF, SPARQL, DBPedia*

REFERENCES

1. RDF Schema 1.1 // W3C. URL: https://www.w3.org/TR/rdf-schema

2. *Tim Berners-Lee*. Linked Data. URL: https://www.w3.org/DesignIssues/LinkedData.html

3. *Amrapali Zaveri*, University of Leipzig. Linked Data Quality of DBpedia, Freebase, OpenCyc, Wikidata, and YAGO. URL: http://www.aifb.kit.edu/ images/c/ca/KG-Comparison-SWJ-Article.pdf

4. Learn about DBPedia. URL: https://wiki.dbpedia.org/about

5. Auto-generated trivia questions based on DBPedia data. URL: https://github.com/n1try/linkeddata-trivia

6. Linked Open Data Seminar 2016 – Knowledge Panel. URL: https://github.com/n1try/kit-lod16-knowledge-panel

7. *G. Vega-Gorgojo, Don Naipe*. Clover quiz: A mobile trivia game based on DBpedia data. URL: http://ceur-ws.org/Vol-1963/paper474.pdf

8. Virtuoso SPARQL Query Editor // DBpedia. URL: https://dbpedia.org/sparql

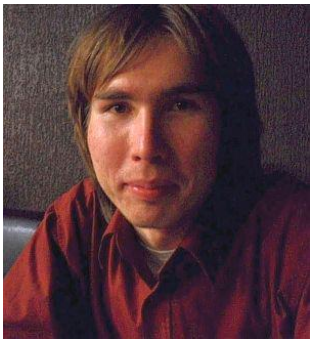
9. Ontology Classes // DBPedia mappings. URL: http://mappings.dbpedia.org/server/ontology/classes/

СВЕДЕНИЯ ОБ АВТОРАХ

АНДРЕИЧЕВ Михаил Дмитриевич – бакалавр Высшей школы информационных технологий и интеллектуальных систем Казанского (Приволжского) федерального университета.

Michail Dmitrievich ANDREICHEV – bachelor of Higher School of ITIS KFU.

email: andreichev.m@mail.ru



ФЕРЕНЕЦ Александр Андреевич – ассистент, преподаватель кафедры программной инженерии Высшей школы информационных технологий и интеллектуальных систем Казанского (Приволжского) федерального университета.

Alexander Andreevich FERENETZ – assistant at Department of Software Engineering of Higher School of ITIS KFU

email: ist.kazan@gmail.com

Материал поступил в редакцию 28 июня 2019 года